Comparison of Real Time System Scheduling Algorithm

1 Priya Yeole, 2 Monika Atey, 3 Ashwini Pathade, 4 Manjusha Badge
1, 2, 3, 4 Student, Department of Computer Technology, RGCER, Nagpur-441110, Maharashtra, India

Abstract - Real time system scheduling is the way used for managing the order of the job which is performed by a CPU of a computer. The goal of scheduling is to engage the CPU at its maximum capacity limit and no process shall wait for longer time period and to finish the entire task in small possible time. In this paper, we discuss various types of Scheduling algorithms and Compare their performance on terms of throughput and waiting period. The four basic type of Scheduling are: Round Robin (RR) scheduling, First Come First Served (FCFS), Priority Based scheduling, Shortest Job First (SJF). These entire algorithms scheme have some drawback and have not been optimized yet. In this paper we propose a Median based Time quantum based scheduling algorithm which is combination of SJF & RR where all the jobs in the queue are first aligned as per their burst time in ascending order and then Round robin is applied for improving the performance. We are mainly work on DPZL and FPZL scheduling algorithms.

Keywords - Scheduling, Multiprocessor, Computation Time, Multicore, Multithreaded, FPZL & DPZL.

1. Introduction

Real time system scheduling can be defined as a mechanism of tool to control the execution of number of working and non-working processes perform work by a computer. CPU is the important of all the resources available in a system that are scheduled before use. The basic idea is to keep the processor busy as much as possible by executing a process, and then switch it to another working process. Each process can be simply classified into Groups based on several properties of the process and permanently assigned to one queue. In the DPZL scheduling, the processes can be dynamically moved in different position queues. So processes that need a large amount of CPU time are sent to the low priority queues and process requiring less amount of CPU and more other bounds are sent to high priority queues.

The Scheduling performance analyzed on following features:

1. **CPU utilization** - The maximum use of CPU when it is busy.
2. **Throughput** - It is the number of processes that complete there execution per unit time.
3. **Turnaround Time** - It is the amount needed for execution of a single process.
4. **Waiting Time** - It is the amount of time a process waits in the ready queue.
5. **Response Time** - This is the amount of time takes from when a request was submitted until the first response is produced not output.

Scheduling can be divided into two categories
1. Non preemptive - A non-preemptive scheduling algorithm gets process to run then it just lets it run until it blocks or until it voluntarily give way by CPU, in other words it bound itself with the first task or job until Unless finished, for e.g. DPZL, FPZL.
2. Preemptive - In this type of scheduling execution of process can be preempted before the completion of the task.

Fig 1. The Basic attributes of task

2. Proposed System

2.1 Scheduling on Processor

Processor platforms include one processor on which number of jobs can be executed. In processor scheduling,
central state information of the entire task states accurately. In a single processor multi-programming system, multiple processes/tasks are contained within memory. Processes survive between Running, Ready, waiting, Blocked, and Suspend. A main goal is to keep the processor busy, by allocating task to the processor to execute, and always having at least one process able to execute. To keep the processor busy is the main purpose of process scheduling. Processor scheduling is categorized as follows:

- **Long-term scheduling**: To add to the processes that are fully or partially in memory.
- **Short-term scheduling**: The decisions as to which process to execute next or in future.

The goal of the scheduling in processor is to achieve high throughput, high processor utilization, number of processes completed per unit time and low response time. But processor scheduling affects the performance of the system, because it determines which process will wait and which will progress.

### 2.2 Scheduling on Multiprocessors

Multiprocessors platforms include more than one processor on which jobs can get executed. The approaches to multiprocessor real-time scheduling can be categorized into two classes: partitioned and global. Under partitioning, the set of tasks is statically partitioned among.

- The only completely accurate way of direction to evaluate a scheduling algorithm scheme is to code it up, put it in the any operating system and see how it works.
- The algorithm is add to the test in a real system under of real operating conditions.
- The main difficulty with this approach is high cost.
- The algorithm has to be coded and the operating system has to be modified to support it.
- Another problems that the arch structure in which the algorithm is used will change.
- New programs will be written and new kinds of problems with be handled.

### 2.3 FPZL (Fixed Priority Until Zero Scheduling)

FPZL is a non-preemptive scheduling algorithm. It uses First in- First out) strategy to assign the priority to processes in the order, that is same as the request made by process for the processor. The process or job that requests the CPU first is allocated the CPU first and other if in the queue has to wait until the CPU is free. All the later arriving jobs are inserted into the tail (rear) of the ready queue and the process to be executed next is removed from the head (front) of the queue and the control of current process is transferred to the CPU.

Gantt chart for above process as per FCFS is:

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
<th>P10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>34</td>
<td>57</td>
<td>68</td>
<td>134</td>
<td>155</td>
<td>211</td>
<td>227</td>
<td>236</td>
<td>253</td>
</tr>
</tbody>
</table>

Average waiting time= 1375/10=137.5ms  
Turnaround time =burst time +waiting time  
Average turnaround time = 1657/10=165.7 ms  

**Fig. 2**: Gantt chart of FCFS

### 2.4 DPZL (Dynamic Priority until Zero Laxity)

There are many situations where some portions of tasks are very important which is to be scheduled on a priority...
basis than the other portions of the task. In this algorithm the task is logically divided into two subtasks, mandatory and optional. Using this concept the mandatory portion of every task will be executed first and later on the optional portions will be executed. It will take input of tasks containing computation time, mandatory portion, optional portion, deadline, and period. As well as some input for processor including Number of processors, period and capacity. The task which will utilize the system more is assigned highest priority and accordingly the priorities are assigned. Schedule tasks according to these assigned priorities. The mandatory tasks will be executed first and then the optional tasks will be executed. While executing mandatory tasks if any zero laxity task arrives, then give that zero laxity task a higher priority than the task which was currently executing and add that pre-empted task to the ready queue and allocate the processor to the zero laxity tasks till it completely gets executed. Capabilities and the communication capabilities of the resources, same as. But the job groups are sent to the corresponding resources based on Largest Job First (LJF) strategy and the results of the processing are sent back to the user after they have been computed at their respective resources. The main principle behind the bandwidth scheduling is the scheduling priorities taking into consideration the communication capabilities of the resources. This does not considered the dynamic source characteristics into account and the scheduling strategy is not ensuring that the resource having a sufficient bandwidth to send the group jobs within required time same as above approach.

A grouping-based _ne-grained job scheduling model is presented in by Liu and Liao, the grained jobs grouped into forming coarse-grained are allocated to the available resources according to their processing capabilities and network bandwidth in Largest Job First (LJF) order. But here resource are selected in FCFS order, there is no priority for selecting resources.

3. System Architecture

A system architecture or systems model is the conceptual model that defines the structures, behaviors, and different views of a system.
Step-4:

While executing mandatory tasks if any zero laxity task arrives, then give that zero laxity task a higher priority than the task which was currently executing and add that pre-empted task to the ready queue and allocate the processor to the zero laxity tasks till it completely gets executed.

Step-5:

Repeat step 3,4 for optional tasks also.

Step-6:

Schedule tasks from ready queue.

Step-7:

Follow steps 3,4, 5, 6

5. Design Model

The main goal of design is to achieve the comparison between FPZL and DPZL algorithms. So, we categorize our goal into the following:

- **Release/ready time**: The time a task is ready to run and just waits for the scheduler to activate it.
- **Deadline**: The time when a task must be finished executing.
- **Execution/run time**: The active computation time a tasks need to complete.
- **Response time**: The time to finish execution & Measured from release time to execution completes.
- **Priority**: The importance given a task in context of the schedule at hand.

The modules are as follows:

**Home Page**

This is the homepage or first module of our project. In this module enter all the entries mention in the table. Choose the algorithm if you want. Enter the name of task, enter the value of Release Time, Computation Time, Period, Deadline, Core Capacity, Core Period and so on.
Compare FPZL & DPZL:

Compare the FPZL and DPZL algorithm this output is come.

Fig. 6 : Comparison of two algorithms.

6. Conclusions

The motivation for our work was the desire to improve upon current state-of-the-art global scheduling methods for hard real-time systems in terms of practical techniques that enable the efficient use of processing capacity. The intuition behind our work was that dynamic priority scheduling has the potential to schedule many more task sets than fixed task or fixed job priority algorithms, and yet this theoretical advantage has to be tempered by the need to avoid prohibitively large overheads due to a high number of pre-emption. This led us to consider minimally dynamic scheduling algorithms which permit each job to change priority at most once during its execution. One such algorithm is EDZL. We applied the zero-laxity rule from EDZL to global FP scheduling, forming the FPZL scheduling algorithm. The number of context switches with FPZL is at most two per zero-laxity task, and one per ordinary task. As there are at most $m$ zero-laxity tasks, the increase in overheads compared to global FP scheduling is tightly bounded.

The key participation of this paper are as follows:

- The zero-laxity rule employed by FPZL appears to have a large impact on task set schedulability, compared to the Performance of global FP scheduling, as shown by the simulation results. The performance potential of FPZL using DCMPO was found to be broadly similar to that of EDZL, and significantly better than that of global FP or global EDF scheduling.
- Using Ardsley’s OPA algorithm to assign task priorities, the polynomial time schedulability test for FPZL results in a modest improvement over the equivalent test for global.
- FP scheduling in the case of constrained-deadline task sets, with an increased improvement for implicit-deadline task sets.

References


Author’s Notes

Priya Sukhdeo Yeole, Research Scholar in computer Department. Rajiv Gandhi College Of Engineering and Research,RTMNU university she is pursuing Bachelor’s degree in computer technology 2015,from RGCER, Nagpur, MH, and India